**Metering**

**Authors**

* PhD Student Marian Banica
* PhD Student Bogdan Zamfir
* Prof. Ciprian Lupu
* Radu Lupașcu, Student
* Radu Edu, Student (Theoretical High School Videle)
* Andu Alexe, Student (Theoretical High School Videle)
* Bogdan Olteanu, Student (Theoretical High School Videle)

**Summary**

This paper describes the design and implementation of a smart home system for real-time energy monitoring and management using IoT sensors, Arduino (for prototyping), and Siemens PLCs for industrial-grade reliability, integrated with the Ignition SCADA platform. The project’s objective is to monitor, record, and optimize electric consumption at the level of individual sockets in a classroom, with all data being visualized and stored in real-time for further analysis and utility management. The paper details hardware/software integration, problems encountered during development, and proposed solutions, serving as a guide for similar educational or industrial projects.

**Introduction**

With the rapid adoption of digitalization and Industry 4.0 standards, real-time monitoring and intelligent management of energy resources become essential both in homes and industry. This project demonstrates how a modular, scalable energy management system can be built with off-the-shelf IoT components, Arduino microcontrollers, Siemens PLCs, and the Ignition SCADA platform, highlighting key challenges and best practices for integrating educational and industrial environments.

**Objectives**

* To develop a robust energy data acquisition and transmission system using Arduino and/or Siemens S7-1200 PLC.
* To integrate this data into a SCADA platform (Ignition) for visualization, live monitoring, and further analysis.
* To store all readings in an SQL database, allowing for historical tracking and daily reporting.
* To document integration challenges, errors, and solutions for future reference and scalability.

**System Architecture & Methodology**

**Hardware:**

* **Arduino UNO R3** (Bitmi Smart Home kit) for rapid prototyping.
* **Siemens S7-1200 PLC** (Model 6ES7 211-1BE40-0XB0) for industrial-level implementation.
* **IoT Sensors**: DHT (temperature/humidity), servo motors, window/door sensors, MAX485 modules for RS485/Modbus communications.
* **Computation**: Laptop/PC (min. 12GB RAM) with Windows 11, running TIA Portal 19 and VMware Workstation Pro for virtualization.
* **Network**: Ethernet cables, RS232-to-RS485 adapters for reliable communication.

**Software:**

* **Arduino IDE** for code development and sensor interfacing.
* **Siemens TIA Portal v19** for PLC configuration and OPC UA communication.
* **Ignition SCADA + Designer** for system visualization, tag management, and database integration.
* **MySQL** for data storage and query/reporting.

**Implementation Steps:**

1. **Arduino Phase (Prototyping)**
   * Connect sensors and actuators to Arduino, use the Modbus RTU protocol (MAX485) for data transmission.
   * Debug communication using Modbus Tester tools, identify and fix library/documentation issues.
2. **Transition to PLC (Industrialization)**
   * Program and configure the Siemens S7-1200 PLC with TIA Portal, create data blocks for variable storage.
   * Enable OPC UA server on the PLC, configure access rights, licenses, and security.
   * Connect the PLC to Ignition via OPC UA, create tags for each measured value.
3. **SCADA & Database Integration**
   * Set up the Ignition Gateway and Designer, create projects and tags for real-time display.
   * Convert Amps to Watts in the SCADA (we are using ammeters for measuring and the voltage is a constant 220V so we are using the standard formula of Watts = Amps x 220V)
   * Implement SQL database storage for all daily readings, with daily reset and historical queries.
   * Develop scripts and graphs to visualize both live and historical energy consumption (Total Watts, kWh per day).
4. **Testing, Troubleshooting & Optimization**
   * Systematically document all errors: communication timeouts, checksum errors, firmware incompatibilities, VM network configuration issues, etc.
   * Upgrade software versions, reconfigure network and security settings as needed.
   * Validate functionality with test code and simulated sensor data.

**Results**

* **Data Acquisition**: Reliable acquisition and transmission of sensor data (current, temperature, environmental status) via Modbus (Arduino) and OPC UA (PLC).
* **Visualization**: The Ignition interface provides live monitoring of power consumption per channel, live charts, and numeric displays for both Amps and Watts.
* **Database Logging**: All values are stored in MySQL, supporting queries and daily kWh aggregation for reporting and future analysis.
* **Challenges Addressed**:
  + Lack of documentation for Modbus libraries on Arduino led to a shift towards PLCs for industrial viability.
  + Incompatibilities between TIA Portal/PLC firmware versions (solved by upgrading to TIA 19 and running inside a VM).
  + Hardware limitations were identified as the root cause for some reading failures; writing to the PLC remained functional and provided a workaround.
  + SCADA-PLC connection problems (item paths, server interfaces) required specific configuration adjustments and updates.

**Conclusions**

The project successfully demonstrates the integration of IoT sensors, open-source and industrial controllers, and modern SCADA solutions for real-time energy management.  
The migration from Arduino to Siemens PLC reflects both a didactic approach and the real-world constraints faced in industrial automation.  
Ignition SCADA, combined with SQL storage, allows for comprehensive monitoring, analysis, and reporting—creating a foundation for future energy efficiency optimizations or extensions (e.g., to larger buildings, multiple classrooms, or integration with BMS systems).

The documented errors and solutions contribute valuable insight for students, engineers, and practitioners seeking to implement similar systems with minimal downtime and scalable architecture.

A qr code with a white background

AI-generated content may be incorrect.A more detailed documentation on this matter can be found on our GitHub repository: <https://github.com/banica/metering>